Please answer each question in the space provided.

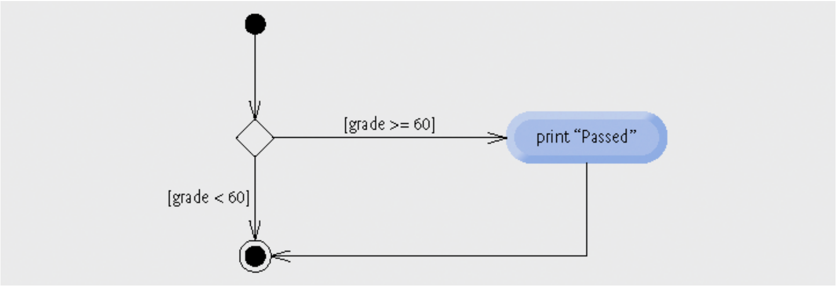
abstract body cast class constructor declared default destroyed encapsulated final friendly garbage import inherit instance instantiated interface members new none null Object one overloading overriding package private protected public static super switch this two while zero

1. **Data and methods may be hidden or encapsulated within a class by specifying the _________ or ___________ visibility modifiers. Members declared ___________ are visible everywhere.**
2. **java.lang.__________ is the default superclass for a class. It is the root of the Java class hierarchy and has no superclass itself. All Java classes inherit the methods defined by this class.**
3. **Method ____________ is the practice of defining multiple methods which have the same name but have different argument lists.**
4. **Method ____________ occurs when a class redefines a method inherited from its superclass.**
5. **From a subclass, you can explicitly invoke an overridden method of a superclass with the __________ keyword.**
6. **Objects are created with the ________ keyword, which invokes a class constructor method with a list of arguments.**
7. **Objects are not explicitly _______________ in any way.**
8. **The Java ____________ collector automatically reclaims objects no longer used.**
9. **If a class does not explicitly define a _______________, Java provides a default one.**
10. **A class may ___________ the non-private methods and variables of another class by "subclassing"--i.e., by declaring that class in its extends clause.**
11. **An ____________________ method has no method body (i.e., no implementation).**
12. **An abstract class contains abstract methods. The methods must be implemented in a subclass before the subclass can be ___________________.**
13. **An ________________ is a collection of abstract methods and constants (static final variables).**
14. **A class implements an interface by declaring the interface in its implements clause and by providing a method ________________ for each of the abstract methods in the interface.**
15. **A _________ is a collection of data and methods that operate on that data.**
16. **An object's fields and methods are known as its ________________ and are accessed with a dot between the object name and their name.**
17. **____________ (also known as non-static) variables occur in each instance of a class.**
18. **____________ (also known as class) variables are associated with the class.**
19. **The number of copies of a class variable is _______ regardless of the number of instances of a class.**
20. **You can declare a class to be ___________ if you don't want there to be any subclasses.**

Give an example of a simple Java program that uses the conditional operator:

What are the three kinds (or categories) of control structures that can be used in a Java program? Give an example (or small section of code) for each.

What does the following UML diagram depict?



For each line, indicate the associativity (RTL=right to left, LTR=left to right)

Operators	Associativity	Type
`++ --`	________	unary postfix
`++ -- + - (` *type* `)`	________	unary prefix
`* / %`	________	Multiplicative
`+ -`	________	Additive
`< <= > >=`	________	Relational
`== !=`	________	Equality
`?:`	________	Conditional
`= += -= *= /= %=`	________	assignment

Consider a Java program which is supposed to investigate the random number generator (provided by the java.util.Random class) to see if it provides a reasonably uniform distribution among the integers (0, 1, 2, … , 9). Examples of the intended output when the program is used to check out the first 100,000 calls to the random number generator (for three separate runs of the program) are:

```
0: 9933                        0: 9985                        0: 9890
1: 9999                        1: 9790                        1: 10005
2: 10019                       2: 9950                        2: 9947
3: 9897                        3: 10063                       3: 10038
4: 10001                       4: 9975                        4: 10085
5: 9883                        5: 10089                       5: 10093
6: 10061                       6: 10057                       6: 10062
7: 10093                       7: 10085                       7: 9973
8: 10053                       8: 9963                        8: 9954
9: 10061                       9: 10043                       9: 9953
```

which shows that the distribution *is* fairly uniform (each digit is produced with approximately the same likelihood), since each of the ten digits is produced approximately one-tenth of the time. Identify and correct any errors in the program (shown below).

```
import java.util.Random();                      // import the class Random

class RandomTest {                              // declare and define our class

   public static void main (String args[]) {    // the main method
      int[] ndigits = new int[10];              // an array for storing the counts
      int n;                                    // the random number (0 - 9)

      Random  myRandom  =   Random();           // construct a new Random object

                                                // Initialize the array:
      for (int i = 0; i++; i < 10) {            // So, for each digit
         ndigits[i] = 0;                        // set the count to zero
      }

      // Test the random number generator a whole lot (100,000 times)
      for (long i=0; i < 100,000; i++) {
         n = myRandom.nextInt(10);              // generate a new random value (0 - 9)
         ndigits[n];++                          // increment the count for that digit
      }
                                                // print the results
      for (int i = 0; i < 10; i++) {
         System.out.println(i + ": " + ndigits[i]);
      }
   }
}
```

______________________________ ANSWERED EXAM FOLLOWS _________________________

Answers / corrections shown in red

Please answer each question in the space provided.

abstract body cast class constructor declared default destroyed encapsulated final friendly garbage import inherit instance instantiated interface members new none null Object one overloading overriding package private protected public static super switch this two while zero

1. **Data and methods may be hidden or encapsulated within a class by specifying the *private* or *protected* visibility modifiers. Members declared *public* are visible everywhere.**
2. **java.lang.*Object* is the default superclass for a class. It is the root of the Java class hierarchy and has no superclass itself. All Java classes inherit the methods defined by this class.**
3. **Method *overloading* is the practice of defining multiple methods which have the same name but have different argument lists.**
4. **Method *overriding* occurs when a class redefines a method inherited from its superclass.**
5. **From a subclass, you can explicitly invoke an overridden method of a superclass with the *super* keyword.**
6. **Objects are created with the *new* keyword, which invokes a class constructor method with a list of arguments.**
7. **Objects are not explicitly *destroyed* in any way.**
8. **The Java *garbage* collector automatically reclaims objects no longer used.**
9. **If a class does not explicitly define a *constructor* Java provides a default one.**
10. **A class may *inherit* the non-private methods and variables of another class by "subclassing"-- i.e., by declaring that class in its extends clause.**
11. **An *abstract* method has no method body (i.e., no implementation).**
12. **An abstract class contains abstract methods. The methods must be implemented in a subclass before the subclass can be *instantiated***
13. **An *interface* is a collection of abstract methods and constants (static final variables).**
14. **A class implements an interface by declaring the interface in its implements clause and by providing a method *body* for each of the abstract methods in the interface.**
15. **A *class* is a collection of data and methods that operate on that data.**
16. **An object's fields and methods are known as its *members* and are accessed with a dot between the object name and their name.**
17. ***instance* (also known as non-static) variables occur in each instance of a class.**
18. ***static* (also known as class) variables are associated with the class.**
19. **The number of copies of a class variable is *one* regardless of the number of instances of a class.**
20. **You can declare a class to be *final* if you don't want there to be any subclasses.**

Give an example of a simple Java program that uses the conditional operator:

```
public class Welcome {
    public static void main (String args[ ]) {
        System.out.println("Expression evaluated to "+
(true?"True":"False"));
    }
}
```

What are the three kinds (or categories) of control structures that can be used in a Java program? Give an example (or small section of code) for each.

Sequential

```
int   k = 2;
double   s = 2.0/3.0;
s += k * s;
```

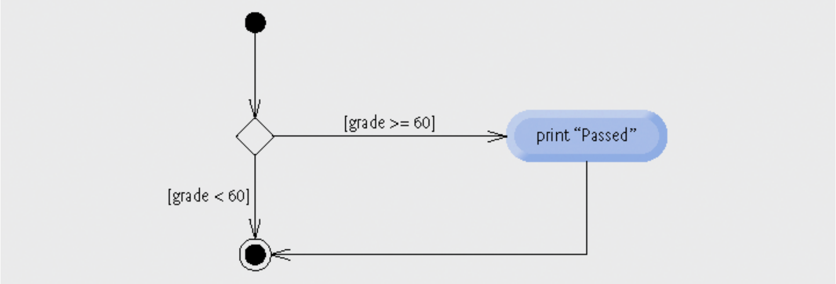
Selection

```
if (k > 1) {
   System.out.print(“ok”);
}
```

Repetition

```
for (k=0; k<5; k++) {
    s+=k;
}
```

What does the following UML diagram depict?



An “if” statement that tests the condition “grade >= 60” and if true prints “Passed”. If the condition is not true (i.e., “grade <60”) , then it does nothing and control is passed to the next statement.

For each line, indicate the associativity (RTL=right to left, LTR=left to right)

Operators	Associativity	Type
++ --	RTL	unary postfix
++ -- + - (*type*)	RTL	unary prefix
* / %	LTR	Multiplicative
+ -	LTR	Additive
< <= > >=	LTR	Relational
== !=	LTR	Equality
?:	RTL	Conditional
= += -= *= /= %=	RTL	assignment

Consider a Java program which is supposed to investigate the random number generator (provided by the java.util.Random class) to see if it provides a reasonably uniform distribution among the integers (0, 1, 2, … , 9). Examples of the intended output when the program is used to check out the first 100,000 calls to the random number generator (for three separate runs of the program) are:

```
0: 9933                 0: 9985                 0: 9890
1: 9999                 1: 9790                 1: 10005
2: 10019                2: 9950                 2: 9947
3: 9897                 3: 10063                3: 10038
4: 10001                4: 9975                 4: 10085
5: 9883                 5: 10089                5: 10093
6: 10061                6: 10057                6: 10062
7: 10093                7: 10085                7: 9973
8: 10053                8: 9963                 8: 9954
9: 10061                9: 10043                9: 9953
```

which shows that the distribution *is* fairly uniform (each digit is produced with approximately the same likelihood), since each of the ten digits is produced approximately one-tenth of the time. Identify and correct any errors in the program (shown below).

```
import java.util.Random;                        // import the class Random

class RandomTest {                              // declare and define our class

   public static void main (String args[]) {    // the main method
      int[] ndigits = new int[10];              // an array for storing the counts
      int n;                                    // the random number (0 - 9)

      Random myRandom = new Random();           // construct a new Random object

                                                // Initialize the array:
      for (int i = 0; i < 10; i++) {            // So, for each digit
         ndigits[i] = 0;                        // set the count to zero
      }

      // Test the random number generator a whole lot (100,000 times)
      for (long i=0; i < 100000; i++) {
         n = myRandom.nextInt(10);              // generate a new random value (0 - 9)
         ndigits[n]++;                          // increment the count for that digit
      }
                                                // print the results
      for (int i = 0; i < 10; i++) {
         System.out.println(I + ": " + ndigits[i]);
      }
   }
}
```